EV205823097

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Improved Weil and Tate Pairing Techniques Using Parabolas

Inventor(s):
Anne Kirsten Eisentraeger
Kristin E. Lauter
Peter L. Montgomery

ATTORNEY'S DOCKET NO. MS1-1275US

RELATED PATENT APPLICATIONS

This patent application is related to co-pending patent application ___/_______ (Attorney's Docket No. MS1-1276US), titled "Squared Weil and Tate Pairing Techniques for use with Elliptic Curves", and which is hereby incorporated by reference herein.

TECHNICAL FIELD

This invention relates to cryptography, and more particularly to methods and apparati that implement improved processing techniques for Weil and Tate pairings and other like pairings using parabolas.

BACKGROUND

As computers have become increasingly commonplace in homes and businesses throughout the world, and such computers have become increasingly interconnected via networks (such as the Internet), security and authentication concerns have become increasingly important. One manner in which these concerns have been addressed is the use of a cryptographic technique involving a key-based cipher. Using a key-based cipher, sequences of intelligible data (typically referred to as plaintext) that collectively form a message are mathematically transformed, through an enciphering process, into seemingly unintelligible data (typically referred to as ciphertext). The enciphering can be reversed, allowing recipients of the ciphertext with the appropriate key to transform the ciphertext back to plaintext, while making it very difficult, if not nearly impossible, for those without the appropriate key to recover the plaintext.

Public-key cryptographic techniques are one type of key-based cipher. In public-key cryptography, each communicating party has a public/private key pair. The public key of each pair is made publicly available (or at least available to others who are intended to send encrypted communications), but the private key is kept secret. In order to communicate a plaintext message using encryption to a receiving party, an originating party encrypts the plaintext message into a ciphertext message using the public key of the receiving party and communicates the ciphertext message to the receiving party. Upon receipt of the ciphertext message, the receiving party decrypts the message using its secret private key, and thereby recovers the original plaintext message.

The RSA (Rivest-Shamir-Adleman) method is one well-known example of public/private key cryptology. To implement RSA, one generates two large prime numbers p and q and multiplies them together to get a large composite number N, which is made public. If the primes are properly chosen and large enough, it will be practically impossible (i.e., computationally infeasible) for someone who does not know p and q to determine them from knowing only N. However, in order to be secure, the size of N typically needs to be more than 1,000 bits. In some situations, such a large size makes the numbers too long to be practically useful.

One situation is found in authentication, which can be required anywhere a party or a machine must prove that it is authorized to access or use a product or service. An example of such a situation is in a product ID system for a software program(s), where a user must hand-enter a product ID sequence stamped on the outside of the properly licensed software package as proof that the software has been properly paid for. If the product ID sequence is too long, then it will be cumbersome and user unfriendly.

Additionally, not only do software manufacturers lose revenue from unauthorized copies of their products, but software manufacturers also frequently provide customer support, of one form or another, for their products. In an effort to limit such support to their licensees, customer support staffs often require a user to first provide the product ID associated with his or her copy of the product for which support is sought as a condition for receiving support. Many current methods of generating product IDs, however, have been easily discerned by unauthorized users, allowing product IDs to be generated by unauthorized users.

Given the apparent ease with which unauthorized users can obtain valid indicia, software manufacturers are experiencing considerable difficulty in discriminating between licensees and such unauthorized users in order to provide support to the former while denying it to the latter. As a result, manufacturers often unwittingly provide support to unauthorized users, thus incurring additional and unnecessary support costs. If the number of unauthorized users of a software product is sufficiently large, then these excess costs associated with that product can be quite significant.

New curve-based cryptography techniques have recently been employed to allow software manufacturers to appreciably reduce the incidence of unauthorized copying of software products. For example, product IDs have been generated using elliptic curve cryptographic techniques. The resulting product IDs provide improved security. Curve-based cryptographic techniques may also be used to perform other types of cryptographic services.

As curve-based cryptosystems grow in popularity, it would be useful to have new and improved techniques for performing the computations associated with the requisite mathematical operations. Hence, there is a continuing need for improved mathematical and/or computational methods and apparati in curve-based cryptosystems.

SUMMARY

In accordance with certain exemplary aspects of the present invention, various methods and apparati are provided for use in curve-based cryptosystems.

For example, methods and apparati are provided for use in cryptographically processing information based on elliptic and other like curves. The methods and apparati allow pairings, such as, for example, Weil pairings, Tate Pairings, Squared Weil pairings, Squared Tate pairings, and/or other like pairings to be determined based on algorithms that utilize a parabola. The methods and apparati represent an improvement over conventional algorithms since they tend to be more computationally efficient.

Thus, for example, the above-stated needs and/or others are met by a method for use in curve-based cryptographic logic. The method includes determining at least one curve for use in cryptographically processing selected information, and determining pairings for use in cryptographically processing the selected information by selectively using at least one parabola associated with the curve. In certain implementations, the curve includes an elliptic curve and the pairings may include Weil pairings, Squared Weil pairings, Tate pairings, Squared Tate pairings, and/or other like pairings.

The method may also include cryptographically processing the selected information based on the pairings. This may include encrypting and/or decrypting the selected information and outputting corresponding processed information. The cryptographic process may include a key-based process, an identity-based encryption process, a product identification (ID)-based process, a short signature-based process, or the like.

In certain implementations, determining the pairings may also include determining at least a first function and a second function that share a point on the elliptic curve, determining the parabola that is associated with the shared point, and a first line and a second line associated with the parabola, determining a third function based on the first line and the second line, and determining the pairings based on the third function.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings. The same numbers are used throughout the figures to reference like components and/or features.

Fig. 1 is a block diagram illustrating an exemplary cryptosystem in accordance with certain implementations of the present invention.

Fig. 2 illustrates an exemplary system using a product identifier to validate software in accordance with certain implementations of the present invention.

Figs. 3a-b illustrate exemplary processes for use in curve-based cryptosystems in accordance with certain implementations of the present invention.

Fig. 4 illustrates a more general exemplary computer environment which can be used in various implementations of the invention.

DETAILED DESCRIPTION

Introduction

The discussions herein assume a basic understanding of cryptography by the reader. For a basic introduction of cryptography, the reader is directed to a book written by Bruce Schneier and entitled "Applied Cryptography: Protocols, Algorithms, and Source Code in C," published by John Wiley & Sons with copyright 1994 (or second edition with copyright 1996).

Described herein are techniques that can be used with a curve-based cryptosystem, and in particular elliptic curve-based cryptosystems. In certain examples, the techniques take the form of methods and apparati that can be implemented in logic within one or more devices. One such device, for example, is a computing device that is configured to perform at least a portion of the processing required for a particular cryptographic capability or application.

The techniques provided herein can be implemented and/or otherwise adapted for use in a variety of cryptographic capabilities and applications. By way of example, the techniques may be employed to support: key generation logic, e.g., for one-round three-way key establishment applications; identity-based encryption logic; short signature logic, e.g., product identifier logic; and/or other like cryptographic logic.

The term logic as used herein is meant to include any suitable form of logic that may be employed. Thus, for example, logic may include hardware, firmware, software, or any combination thereof.

The term curve-based cryptosystem as used herein refers to logic that at least partially provides for curve-based signature generation and verification using key(s) that are generated based at least partially on aspects or characteristics of an elliptic curve or other like curve.

Such curve-based cryptosystems can be used to encrypt any of a wide variety of information. Here, for example, one exemplary cryptosystem is described primarily with respect to generation of a short signature or product identifier, which is a code that allows validation and/or authentication of a machine, program, user, etc. The signature is a "short" signature in that it uses a relatively small number of characters.

With this in mind, attention is drawn to Fig. 1, which is a block diagram illustrating an exemplary cryptosystem 100 in accordance with certain implementations of the present invention. Cryptosystem 100 includes an encryptor 102 and a decryptor 104. A plaintext message 106 is received at an input module 108 of encryptor 102, which is a curve-based encryptor that encrypts message 106 based on a public key generated based on a secret known by decryptor 104. Plaintext message 106 is typically an unencrypted message, although encryptor 102 can encrypt any type of message/data. Thus, message 106 may alternatively be encrypted or encoded by some other component (not shown) or a user.

An output module 110 of encryptor 102 outputs the encrypted version of plaintext message 106, which is ciphertext 112. Ciphertext 112 can then be communicated to decryptor 104, which can be implemented, for example, on a computer system remote from a computer system on which encryptor 102 is implemented. Given the encrypted nature of ciphertext 112, the communication link between encryptor 102 and 104 need not be secure (it is typically presumed that the communication link is not secure). The communication link can be any of a wide variety of public and/or private networks implemented using any of a wide variety of conventional public and/or proprietary protocols, and including both wired and wireless implementations. Additionally, the communication link may include other non-computer network components, such as hand-delivery of media including ciphertext or other components of a product distribution chain.

Decryptor 104 receives ciphertext 112 at input module 114 and, being aware of the secret used to encrypt message 106, is able to readily decrypt ciphertext 112 to recover the original plaintext message 106, which is output by output module 116 as plaintext message 118. Decryptor 104 is a curve-based decryptor that decrypts the message based on the same curve as was used by encryptor 102.

Encryption and decryption are performed in cryptosystem 100 based on a secret, such as points on the elliptic curve. This secret is known to decryptor 104, and a public key generated based on the secret is known to encryptor 102. This knowledge allows encryptor 102 to encrypt a plaintext message that can be decrypted only by decryptor 104. Other components, including encryptor 102, which do not have knowledge of the secret cannot decrypt the ciphertext (although decryption may be technically possible, it is not computationally feasible). Similarly, decryptor 104 can also generate a message using the secret and based on a plaintext message, a process referred to as digitally signing the plaintext message. This signed message can then be communicated to other components, such as encryptor 102, which can in turn verify the digital signature based on the public key.

Fig. 2 illustrates an exemplary system using a product identifier to validate software in accordance with certain implementations of the present invention. Fig.

2 illustrates a software copy generator 120 including a product identifier (ID) generator 122. Software copy generator 120 produces software media 124 (e.g., a CD-ROM, DVD (Digital Versatile Disk), etc.) that contains typically all the files needed to collectively implement a complete copy of one or more application programs, (e.g., a word processing program, a spreadsheet program, an operating system, a suite of programs, and so forth). These files are received from source files 126, which may be a local source (e.g., a hard drive internal to generator 120), a remote source (e.g., coupled to generator 120 via a network), or a combination thereof. Although only a single generator 120 is illustrated in Fig. 2, typically multiple such generators operate individually and/or cooperatively to increase the rate at which software media 124 can be generated.

Product ID generator 122 generates a product ID 128 that can include numbers, letters, and/or other symbols. Generator 122 generates product ID 128 using the curve-based encryption techniques described herein. The product ID 128 is typically printed on a label and affixed to either a carrier containing software media 124 or a box into which software media 124 is placed. Alternatively, the product ID 128 may be made available electronically, such as a certificate provided to a user when receiving a softcopy of the application program via an on-line source (e.g., downloading of the software via the Internet). The product ID can serve multiple functions. First, the product ID can be cryptographically validated in order to verify that the product ID is a valid product ID (and thus allowing, for example, the application program to be installed). Additionally, the product ID can optionally serve to authenticate the particular software media 124 to which it is associated.

The generated software media 124 and associated product ID 128 are then provided to a distribution chain 130. Distribution chain 130 represents any of a variety of conventional distribution systems and methods, including possibly one or more "middlemen" (e.g., wholesalers, suppliers, distributors, retail stores (either on-line or brick and mortar), etc.). Regardless of the manner in which media 124 and the associated product ID 128 are distributed, eventually media 124 and product ID 128 are purchased (e.g., licensed), by the user of a client computer 132.

Client computer 132 includes a media reader 134 capable of reading software media 124 and installing the application program onto client computer 132 (e.g., installing the application program on to a hard disk drive (not shown) of client computer 132). Part of this installation process involves entry of the product ID 128. This entry may be a manual entry (e.g., the user typing in the product ID via a keyboard), or alternatively an automatic entry (e.g., computer 132 automatically accessing a particular field of a license associated with the application program and extracting the product ID there from). Client computer 132 also includes a product ID validator 136 which validates, during installation of the application program, the product ID 128. This validation is performed using the curve-based decryption techniques.

If validator 136 determines that the product ID is valid, then an appropriate course of action is taken (e.g., an installation program on software media 124 allows the application to be installed on computer 132). However, if validator 136 determines that the product ID is invalid, then a different course of action is taken (e.g., the installation program terminates the installation process preventing the application program from being installed).

Product ID validator 136 also optionally authenticates the application program based on the product ID 128. This authentication verifies that the product ID 128 entered at computer 132 corresponds to the particular copy of the application being accessed. The authentication can be performed at different times, such as during installation, or when requesting product support or an upgrade. Alternatively, this authentication may be performed at a remote location (e.g., at a call center when the user of client computer 132 calls for technical support, the user may be required to provide the product ID 128 before receiving assistance).

If the application program manufacturer desires to utilize the authentication capabilities of the product ID, then the product ID generated by generator 122 for each copy of an application program is unique. This uniqueness is created by assigning a different initial number or value to each copy of the application program. This initial value can then be used as a basis for generating the product ID.

The unique value associated with the copy of the application program can be optionally retained by the manufacturer as an authentication record 138 (e.g., a database or list) along with an indication of the particular copy of the application program. This indication can be, for example, a serial number embedded in the application program or on software media 124, and may be hidden in any of a wide variety of conventional manners.

Alternatively, the individual number itself may be a serial number that is associated with the particular copy, thereby allowing the manufacturer to verify the authenticity of an application program by extracting the initial value from the product ID and verifying that it is the same as the serial number embedded in the application program or software media 124.

Appropriate action can be taken based on whether the product ID is authenticated. These actions can vary, depending on the manufacturer's desires and/or action being taken at computer 132 that caused the authentication check to occur. For example, if a user is attempting to install an application program then installation of the program may be allowed only if the authentication succeeds. By way of another example, the manufacturer's support technicians may provide assistance to a user of computer 132 only if the authentication succeeds, or an upgrade version of the application program may be installed only if authentication of the previous version of the application program succeeds.

The logic of certain curve-based cryptosystems utilizes what are commonly referred to as "Weil and Tate pairings" during the encryption and/or decryption process when using elliptic curves. The Weil and Tate pairings have been proposed for use in many aspects of cryptography. They may be used, for example, to form efficient protocols to do one-round three-way key establishment, identity-based encryption, short signatures, and the like.

It is important, however, given the amount of processing to have efficient implementations of the Weil and Tate pairings to cut down on the cost of implementing these protocols. Computation of the Weil or Tate pairing in conventional cryptosystems typically follows "Miller's algorithm", which is described, for example, in "Identity-Based Encryption From The Weil Pairing", by Dan Boneh and Matthew Franklin, published in SIAM J. of Computing, Vol. 32, No. 3, pp. 586-615, 2003.

As is well-known, for a fixed natural number m, the Weil pairing e_m is a bilinear map that takes as input two m-torsion points on an elliptic curve, and outputs an m^{th} root of unity. For elliptic curves, as is well-known, the Tate pairing is related to the Weil pairing by the fact that the Weil pairing is a quotient of the output of two applications of the Tate pairing. The algorithms for these pairings depend on constructing rational functions with prescribed patterns of poles and zeros.

The Miller algorithm as typically implemented in conventional curve-based cryptosystems calls for the evaluation of the Weil or Tate pairing by evaluating a function at two selected points on the elliptic curve, wherein one of the points is a "random" point selected using a randomly generated input.

The improved techniques described herein provide increased efficiency and an alternative method to the standard methods which have been proposed. For example, in accordance with certain aspects of the present invention, the improved techniques employ parabolas to help define Weil and/or Tate pairings.

By way of further reference, other exemplary curve-based cryptosystems are provided in the following references: "Short Signatures from the Weil Pairing", by Dan Boneh, et al., in *Advances in Cryptography – Asiacrypt 2001*, Lecture Notes in Computer Science, Vol. 2248, Springer-Verlag, pp. 514-532; and, "The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems (Survey)", by Antoine Joux, in *Algorithmic Number Theory, 5th International Symposium ANTS-V, Sydney, Australia, July 2002 proceedings, Claus Fieker and David R. Kohel (Eds.)*, Lecture Notes in Computer Science, Vol. 2369, Springer-Verlag, pp. 20-32.

Attention is now drawn to Fig. 3a, which is a flow diagram illustrating an exemplary process 150 for use in comparing the Weil and Tate pairings for elliptic curves. In act 152, an addition chain, addition-subtraction chain, or the like, is formed for m, wherein m is an integer greater than zero and an m-torsion point $\boldsymbol{P}$ is fixed on an elliptic curve E. In act 154, $((j+k)\boldsymbol{P}, f_{j+k,\boldsymbol{P}}(\boldsymbol{X}))$ is determined using $(j\boldsymbol{P}, f_{j,\boldsymbol{P}}(\boldsymbol{X}))$ and $(k\boldsymbol{P}, f_{j,\boldsymbol{P}}(\boldsymbol{X}))$, wherein j and k are integers, $j\boldsymbol{P}$, $k\boldsymbol{P}$ and $(j+k)\boldsymbol{P}$ are multiples of point $\boldsymbol{P}$ and $f_{j,\boldsymbol{P}}(\boldsymbol{X})$, $f_{k,\boldsymbol{P}}(\boldsymbol{X})$ and $f_{j+k,\boldsymbol{P}}(\boldsymbol{X})$ are functions in the indeterminate $\boldsymbol{X}$, and $((j+k)\boldsymbol{P}, f_{j+k,\boldsymbol{P}}(\boldsymbol{X}))$ represents an iterative building block for forming the output of the pairing via a chain. With the Weil pairing, for example, $((j+k)\boldsymbol{P}, f_{j+k,\boldsymbol{P}}(\boldsymbol{X}))$ can also be run with $\boldsymbol{P}$ replaced by another m-torsion point $\boldsymbol{Q}$, i.e., $((j+k)\boldsymbol{Q}, f_{j+k,\boldsymbol{Q}}(\boldsymbol{X}))$. In act 156, h_{j+k} is determined given h_j and h_k, wherein h_j, h_k and h_{j+k} are field elements and for example,

$$h_j = f_{j,,\boldsymbol{P}}(\mathbf{Q}_1) \,/\, f_{j,,\boldsymbol{P}}(\mathbf{Q}_2)$$

for certain points $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ (independent of j) on E and the goal is to compute h_m. In conventional Miller algorithms $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ are random value inputs.

In accordance with certain further aspects of the present invention, an improvement is made to act 154 wherein a parabola is introduced for computing Weil pairings, Tate pairings, Squared Weil pairings and/or Squared Tate pairings in a manner that reduces the number of computation steps required. Weil and Tate pairings are well known. Exemplary techniques for determining Squared Weil pairings and Squared Tate pairings are described in the following section and are the subject of co-pending U.S. patent application ___/________ (Attorney's Docket No. MS1-1276US).

Squared Weil Pairing for Elliptic Curves

This section describes Squared Weil pairing, which has the advantage of being more efficient to compute than Miller's algorithm for the original Weil pairing.

The improved algorithm presented herein has the advantage that it is guaranteed to output the correct answer since it does not depend on inputting a randomly chosen m-torsion point. Certain conventional implementations of Miller's algorithm sometimes require multiple iterations of the algorithm, since the randomly chosen m-torsion point may cause the algorithm to fail at times.

Let E: $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ be an elliptic curve over a field K. Introducing some further notation, let:

id be the point at infinity on E;

P, ***Q***, ***R***, ***X*** be points on E, wherein ***X*** is an indeterminate denoting the (main) independent variable of a function;

x(***X***), y(***X***) be (rational) functions mapping a point ***X*** on E to its (affine) x and y coordinates;

line (***P*****,** ***Q***, ***R***)(***X***) be the equation (linear in x(***X***) and y(***X***)) of the line passing through the three points ***P***, ***Q***, ***R*** on E, which satisfy ***P*** **+** ***Q*** **+** ***R*** = **id**, and wherein when two of ***P***, ***Q***, ***R*** are equal, this is a tangent line.

Note, as used herein, a bolded **+** or **−** operator denotes arithmetic in the elliptic curve group, whereas a normal (non-bolded) + or − operator denotes arithmetic in the field K or in the integers.

Function $f_{j,\mathbf{P}}$ and its Construction

If j is an integer and $\mathbf{P}$ a point on E, then $f_{j,\mathbf{P}}$ and $f_{j,\mathbf{P}}(\mathbf{X})$ will refer to a rational function on E whose divisor of zeros and poles is:

$$(f_{j,\mathbf{P}}) = j(\mathbf{P}) - (j\mathbf{P}) - (j-1)(\mathbf{id}),$$

where parentheses around a point on E indicate that it is being considered formally as a point on E. If $j > 1$ and $\mathbf{P}, j\mathbf{P}$, and $\mathbf{id}$ are distinct, then $f_{j,\mathbf{P}}(\mathbf{X})$ has a j-fold zero at $\mathbf{X} = \mathbf{P}$, a simple pole at $\mathbf{X} = j\mathbf{P}$, a $(j-1)$-fold pole at infinity (i.e., at $\mathbf{X} = \mathbf{id}$), and no other poles or zeros.

The theory of divisors states that $f_{j,\mathbf{P}}$ exists and is unique up to a nonzero scale factor (multiplicative constant). If $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are given, then the quotient $f_{j,\mathbf{P}}(\mathbf{Q}_1) / f_{j,\mathbf{P}}(\mathbf{Q}_2)$ is well-defined unless a division by zero occurs.

When $j = 0$ or $j = 1$, $f_{j,\mathbf{P}}$ can be any nonzero constant.

If one knows $f_{j,\mathbf{P}}$ and $f_{k,\mathbf{P}}$ for two integers j and k, then a simple, well-known, construction gives $f_{-j-k,\mathbf{P}}$. One wants $f_{-j-k,\mathbf{P}}$ to satisfy

$$(f_{-j-k,\mathbf{P}}\ f_{j,\mathbf{P}}\ f_{k,\mathbf{P}}) = (f_{-j-k,\mathbf{P}}) + (f_{j,\mathbf{P}}) + (f_{k,\mathbf{P}}) = 3(\mathbf{id}) - ((-j-k)\mathbf{P}) - (j\mathbf{P}) - (k\mathbf{P}).$$

This will be satisfied if we choose $f_{-j-k,\mathbf{P}}$ so that:

$$f_{-j-k,\mathbf{P}}(\mathbf{X})\ f_{j,\mathbf{P}}(\mathbf{X})\ f_{k,\mathbf{P}}(\mathbf{X})\ \mathrm{line}(j\mathbf{P}, k\mathbf{P}, (-j-k)\mathbf{P})(\mathbf{X}) = \text{constant}.$$

Then repeating this construction on $f_{0,\mathbf{P}}$ and $f_{-j-k,\mathbf{P}}$ gives $f_{j+k,\mathbf{P}}$. The line through $0{*}\mathbf{P} = \mathbf{id}$, $(-j-k)\mathbf{P}$, and $(j+k)\mathbf{P}$ is vertical (i.e., its equation does not reference the y-coordinate). This results in the useful constructions

$$f_{j+k,\mathbf{P}}(\mathbf{X}) = f_{j,\mathbf{P}}(\mathbf{X}) f_{k,\mathbf{P}}(\mathbf{X}) \frac{\mathrm{line}\big(j\mathbf{P}, k\mathbf{P}(-j-k)\mathbf{P}\big)(\mathbf{X})}{\mathrm{line}\big(\boldsymbol{id}, (-j-k)\mathbf{P}, (j+k)\mathbf{P}\big)(\mathbf{X})}$$

$$f_{j-k,\mathbf{P}}(\mathbf{X}) = \frac{f_{j,\mathbf{P}}(\mathbf{X})\mathrm{line}\big(\boldsymbol{id}, j\mathbf{P}, -j\mathbf{P}\big)(\mathbf{X})}{f_{k,\mathbf{P}}(\mathbf{X})\mathrm{line}\big(-j\mathbf{P}, k\mathbf{P}, (j-k)\mathbf{P}\big)(\mathbf{X})}$$

Other possibly useful formulae include:

$f_{j,\boldsymbol{id}}$ = constant;

$f_{j,-\boldsymbol{P}}(\boldsymbol{X}) = f_{j,\boldsymbol{P}}(-\boldsymbol{X})$ * (constant);

If $(\boldsymbol{P} + \boldsymbol{Q} + \boldsymbol{R} = \mathbf{id})$, then:

$$f_{j,\boldsymbol{P}}(\boldsymbol{X})f_{j,\boldsymbol{Q}}(\boldsymbol{X})f_{j,\boldsymbol{R}}(\boldsymbol{X}) = \frac{\text{line}(\boldsymbol{P},\boldsymbol{Q},\boldsymbol{R})(\boldsymbol{X})^j}{\text{line}(j\boldsymbol{P},j\boldsymbol{Q},j\boldsymbol{R})(\boldsymbol{X})}.$$

Squared Weil-Pairing Formula

Let m be an odd prime. Suppose $\boldsymbol{P}$ and $\boldsymbol{Q}$ are m-torsion points on E, with neither being the identity and $\boldsymbol{P}$ not equal to $\pm\boldsymbol{Q}$.

Then

$$\frac{f_{m,\boldsymbol{P}}(\boldsymbol{Q})f_{m,\boldsymbol{Q}}(-\boldsymbol{P})}{f_{m,\boldsymbol{P}}(-\boldsymbol{Q})f_{m,\boldsymbol{Q}}(\boldsymbol{P})} = -e_m(\boldsymbol{P},\boldsymbol{Q})^2$$

where e_m denotes the Weil-pairing.

Exemplary Algorithm for $e_m(\boldsymbol{P}, \boldsymbol{Q})^2$

Fix an odd prime m and the curve E. Given two m-torsion points $\boldsymbol{P}$ and $\boldsymbol{Q}$ on E, one needs to compute $e_m(\boldsymbol{P}, \boldsymbol{Q})^2$.

In accordance with certain exemplary implementations of the present invention, the algorithm includes forming an addition or addition-subtraction chain for m. That is, after an initial 1, every element in the chain is a sum or difference of two earlier elements in the chain, until an m appears. Well-known techniques give a chain of length O(log(m)).

For each j in the addition-subtraction chain, form a tuple

$t_j = [j\boldsymbol{P}, j\boldsymbol{Q}, n_j, d_j]$

such that

$$\frac{n_j}{d_j} = \frac{f_{j,\mathbf{P}}(\mathbf{Q})f_{j,\mathbf{Q}}(-\mathbf{P})}{f_{j,\mathbf{P}}(-\mathbf{Q})f_{j,\mathbf{Q}}(\mathbf{P})}.$$

Keeping the numerator and denominator separate until the end is optional.

To do this, start with $t_1 = [\mathbf{P}, \mathbf{Q}, 1, 1]$. Given t_j and t_k, this procedure gets t_{j+k}:

form elliptic curve sums: $j\mathbf{P} + k\mathbf{P} = (j+k)\mathbf{P}$ and $j\mathbf{Q} + k\mathbf{Q} = (j+k)\mathbf{Q}$;

find line: $\text{line}(j\mathbf{P}, k\mathbf{P}, (-j-k)\mathbf{P})(\mathbf{X}) = c0 + c1*x(\mathbf{X}) + c2*y(\mathbf{X})$;

find line: $\text{line}(j\mathbf{Q}, k\mathbf{Q}, (-j-k)\mathbf{Q})(\mathbf{X}) = \text{c0}' + \text{c1}'*x(\mathbf{X}) + \text{c2}'*y(\mathbf{X})$.

Set:

$n_{j+k} = n_j * n_k * (c0 + c1*x(\mathbf{Q}) + c2*y(\mathbf{Q})) * (c0' + c1'*x(\mathbf{P}) - c2'*y(\mathbf{P}))$

and

$d_{j+k}=d_j * d_k * (c0 + c1*x(\mathbf{Q}) - c2*y(\mathbf{Q})) * (c0' + c1'*x(\mathbf{P}) + c2'*y(\mathbf{P}))$.

A similar construction gives t_{j-k} from t_j and t_k. Observe that the vertical lines through $(j+k)\mathbf{P}$ and $(j+k)\mathbf{Q}$ do not appear in the formulae for n_{j+k} and d_{j+k}, — this is because their contributions from $\mathbf{Q}$ and $-\mathbf{Q}$ (or from $\mathbf{P}$ and $-\mathbf{P}$) are equal. Here $-\mathbf{Q}$ is the complement of $\mathbf{Q}$ and $-\mathbf{P}$ is the complement of $\mathbf{P}$.

When $j + k = m$, one can further simplify this to $n_{j+k} = n_j * n_k$ and $d_{j+k} = d_j * d_k$, since $c2$ and $c2'$ will be zero.

Pseudocode may take the following form, for example:

```
procedure Squared_Weil_Pairing($m$, $\mathbf{P}$, $\mathbf{Q}$)
    issue an error if $m$ is not an odd prime.
    if ($\mathbf{P} = \mathbf{id}$ or $\mathbf{Q} = \mathbf{id}$ or $\mathbf{P} = \pm\mathbf{Q}$) then
        return 1;
    else
        $t_1 = [\mathbf{P}, \mathbf{Q}, 1, 1]$;
        use an addition-subtraction chain to get
            $t_m=[m\mathbf{P}, m\mathbf{Q}, n_m, d_m]$.
        issue an error if $m\mathbf{P}$ or $m\mathbf{Q}$ is not $\mathbf{id}$.
        if ($n_m = 0$ or $d_m = 0$) then
            return 1;
        else
            return $-n_m/d_m$;
        end if;
    end if;
```

When n_m and d_m are nonzero, then the computation

$$\frac{n_m}{d_m} = \frac{f_{m,\boldsymbol{P}}(\boldsymbol{Q}) f_{m,\boldsymbol{Q}}(\boldsymbol{-P})}{f_{m,\boldsymbol{P}}(\boldsymbol{-Q}) f_{m,\boldsymbol{Q}}(\boldsymbol{P})}$$

has been successful, and the output is correct. If, however, some n_m or d_m is zero, then some factor such as $c0 + c1*x(\boldsymbol{Q}) + c2*y(\boldsymbol{Q})$ must have vanished. That line was chosen to pass through $j\boldsymbol{P}$, $k\boldsymbol{P}$, and $(-j-k)\boldsymbol{P}$, for some j and k.

This factor does not vanish at any other point on the elliptic curve. Therefore this factor can vanish only if $\boldsymbol{Q} = j\boldsymbol{P}$ or $\boldsymbol{Q} = k\boldsymbol{P}$ or $\boldsymbol{Q} = (-j-k)\boldsymbol{P}$ for some j and k. In all of these cases $\boldsymbol{Q}$ will be a multiple of $\boldsymbol{P}$, ensuring that

$$e_m(\boldsymbol{P}, \boldsymbol{Q}) = 1.$$

Squared Tate Pairing For Elliptic Curves

Squared Tate Pairing Formula

Let m be an odd prime. Suppose $\boldsymbol{P}$ is an m-torsion point on E, and $\boldsymbol{Q}$ is a point on the curve, with neither being the identity and $\boldsymbol{P}$ not equal to a multiple of $\boldsymbol{Q}$. Assume that E is defined over K, where K has $q = p^n$ elements and suppose m divides $q-1$. Then

$$\left(\frac{f_{m,\boldsymbol{P}}(\boldsymbol{Q})}{f_{m,\boldsymbol{P}}(\boldsymbol{-Q})}\right)^{\frac{q-1}{m}} = v_m(\boldsymbol{P}, \boldsymbol{Q})$$

where v_m denotes the squared Tate-pairing.

Exemplary Algorithm For $v_m(\boldsymbol{P}, \boldsymbol{Q})$

Fix an odd prime m and the curve E. Given an m-torsion point $\boldsymbol{P}$ on E and a point $\boldsymbol{Q}$ on E, one needs to compute $v_m(\boldsymbol{P}, \boldsymbol{Q})$.

As before, one starts with an addition or addition-subtraction chain for m.

For each j in the addition-subtraction chain, one then forms a tuple

$$t_j = [j\boldsymbol{P}, n_j, d_j]$$

such that

$$\frac{n_j}{d_j} = \frac{f_{j,\boldsymbol{P}}(\boldsymbol{Q})}{f_{j,\boldsymbol{P}}(-\boldsymbol{Q})}$$

Keeping the numerator and denominator separate until the end is optional.

Start with $t_1 = [\boldsymbol{P}, 1, 1]$. Given t_j and t_k, to get t_{j+k}:

form the elliptic curve sum $j\boldsymbol{P} + k\boldsymbol{P} = (j+k)\boldsymbol{P}$;

find $\text{line}(j\boldsymbol{P}, k\boldsymbol{P}, (-j-k)\boldsymbol{P})(\boldsymbol{X}) = c0 + c1*x(\boldsymbol{X}) + c2*y(\boldsymbol{X})$;

set:

$$n_{j+k} = n_j * n_k * (c0 + c1*x(\boldsymbol{Q}) + c2*y(\boldsymbol{Q}))$$

and

$$d_{j+k} = d_j * d_k * (c0 + c1*x(\boldsymbol{Q}) - c2*y(\boldsymbol{Q})).$$

A similar construction gives t_{j-k} from t_j and t_k. Observe that the vertical lines through $(j+k)\boldsymbol{P}$ and $(j+k)\boldsymbol{Q}$ do not appear in the formulae for n_{j+k} and d_{j+k}, because the contributions from $\boldsymbol{Q}$ and $-\boldsymbol{Q}$ are equal. When $j+k=m$, one can further simplify this to:

$$n_{j+k} = n_j * n_k \text{ and } d_{j+k} = d_j * d_k,$$

since $c2$ will be zero.

When n_m and d_m are nonzero, then the computation

$$\frac{n_m}{d_m} = \frac{f_{m,\boldsymbol{P}}(\boldsymbol{Q})}{f_{m,\boldsymbol{P}}(-\boldsymbol{Q})}$$

has been successful, and after raising to the $(q-1)/m$ power, one will have the correct output. If, however, some n_m or d_m is zero, then some factor such as $c0 + c1*x(\boldsymbol{Q}) + c2*y(\boldsymbol{Q})$ must have vanished. That line was chosen to pass through $j\boldsymbol{P}$, $k\boldsymbol{P}$, and $(-j-k)\boldsymbol{P}$, for some j and k. It does not vanish at any other point on the elliptic curve. Therefore this factor can vanish only if $\boldsymbol{Q} = j*\boldsymbol{P}$ or $\boldsymbol{Q} = k*\boldsymbol{P}$ or $\boldsymbol{Q} = (-j-k)\boldsymbol{P}$ for some j and k. In all of these cases $\boldsymbol{Q}$ will be a multiple of $\boldsymbol{P}$.

Determining Weil and Tate Pairing For Elliptic Curves Using Parabolas

In this section improved techniques in accordance with certain aspects of the present invention are described for computing the Weil pairing, $e_m(\boldsymbol{P},\boldsymbol{Q})$, the Tate pairing, the Squared Weil pairing, and/or the Squared Tate pairing. The improved techniques essentially merge two computation steps for the Weil pairing or the Tate pairing and employ a simpler way to compute the result by using parabolas. The resulting improved algorithm has the advantage of being more computationally efficient than Miller's algorithm for the original Weil pairing.

In this section, let:

E: $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ be an elliptic curve over a field K;

To compute the Weil Pairing, the Tate pairing or the squared Weil or squared Tate pairing one needs to compute $f_{m,\boldsymbol{P}}$ for one or more points $\boldsymbol{P}$ on the curve E. This can be accomplished, for example, as described in the Boneh-Franklin article referenced above and/or using the above constructions/identities and addition-subtraction chains.

Construction of $f_{2j+k,\boldsymbol{P}}$ from $f_{j,\boldsymbol{P}}$ and $f_{k,\boldsymbol{P}}$:

Consider the general elliptic curve given by the equation:

$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$

Suppose one is given $f_{j,\boldsymbol{P}}$ and $f_{k,\boldsymbol{P}}$ and needs to compute $f_{2j+k,\boldsymbol{P}}$. One method computes $f_{2j,\boldsymbol{P}}$ and $f_{2j+k,\boldsymbol{P}}$ by successive applications of the formula:

$$f_{j+k,\boldsymbol{P}}(\boldsymbol{X}) = f_{j,\boldsymbol{P}}(\boldsymbol{X}) f_{k,\boldsymbol{P}}(\boldsymbol{X}) \frac{\text{line}\big(j\boldsymbol{P},k\boldsymbol{P},(-j-k)\boldsymbol{P}\big)(\boldsymbol{X})}{\text{line}\big(\boldsymbol{id},(-j-k)\boldsymbol{P},(j+k)\boldsymbol{P}\big)(\boldsymbol{X})}$$

In accordance with certain implementations of the improved technique, one essentially combines the two line forming steps into one parabola forming step by constructing a parabola going through the points $j\boldsymbol{P}$, $j\boldsymbol{P}$, $k\boldsymbol{P}$, $-2j\boldsymbol{P}-k\boldsymbol{P}$.

To form $f_{2j+k,\boldsymbol{P}}$ one can form $f_{j+k,\boldsymbol{P}}$ followed by $f_{j+k+j,\boldsymbol{P}} = f_{2j+k,\boldsymbol{P}}$.

To compute $f_{j+k,\boldsymbol{P}}$, one finds a line through $j\boldsymbol{P}=(x_1,y_1)$, $k\boldsymbol{P}=(x_2,y_2)$ and $-j\boldsymbol{P}-k\boldsymbol{P} = (x_3, -a_1x_3 - a_3 - y_3)$. Note that the complement of the point $-j\boldsymbol{P}-k\boldsymbol{P}$ is $j\boldsymbol{P}+k\boldsymbol{P} = (x_3, y_3)$, because one is considering elliptic curves of the most general form. Let this line have slope λ_l, where

$$\lambda_1 = \frac{y_1 - y_2}{x_1 - x_2} = \frac{y_1 - (-y_3 - a_3 - a_1x_3)}{x_1 - x_3}$$

and let

$\text{line}_1(\boldsymbol{X}) := y(\boldsymbol{X}) - (-a_1x_3 - a_3 - y_3) - \lambda_1\,(x(\boldsymbol{X}) - x_3).$

To form $f_{j+k+j,\boldsymbol{P}}$ from $f_{j,\boldsymbol{P}}$ and $f_{j+k,\boldsymbol{P}}$ one needs to find a second line, here line_2, through the points $j\boldsymbol{P}$, $(j+k)\boldsymbol{P}$ and $(-j-(k+j))\boldsymbol{P} = -(2j+k)\boldsymbol{P} = (x_4,y_4)$.

Let line_2 have slope λ_2, where

$$\lambda_2 = \frac{y_1 - y_4}{x_1 - x_4} = \frac{y_1 - y_3}{x_1 - x_3}$$

Then line_2 has the form: $\text{line}_2(\boldsymbol{X}) := y(\boldsymbol{X}) - y_3 - \lambda_2(x(\boldsymbol{X}) - x_3)$.

To obtain $f_{2j+k,\boldsymbol{P}}$ in one step, one can form

$$f_{2j+k,\boldsymbol{P}} = f_{j,\boldsymbol{P}} f_{k,\boldsymbol{P}} f_{j,\boldsymbol{P}} \frac{\text{line}_1 * \text{line}_2}{(x-x_3)*(x-x_4)}$$

One may make this more efficient by replacing

$$\frac{\text{line}_1 * \text{line}_2}{(x-x_3)}$$

with the (possibly degenerate) parabola through $j\boldsymbol{P}, j\boldsymbol{P}$, $k\boldsymbol{P}$, and $(-2j-k)\boldsymbol{P}$ which is given by the equation

$$\begin{aligned}\text{parab}(\boldsymbol{X}) &:= (x(\boldsymbol{X}) - x_1)(x(\boldsymbol{X}) + x_1 + x_3 + a_2 + \lambda_1 \lambda_2) \\ &\qquad + (\lambda_1 + \lambda_2 + a_1)(y_1 - y(\boldsymbol{X})) \\ &= (x(\boldsymbol{X}) - x_1)(x(\boldsymbol{X}) + x_1 + x_3 + a_2 + \lambda_1 \lambda_2) \\ &\qquad + (\lambda_1 + \lambda_2 + a_1)y_1 - (\lambda_1 + \lambda_2 + a_1)\, y(\boldsymbol{X}).\end{aligned}$$

For the Weil pairing, the Tate pairing and the squared Weil or squared Tate pairing, one may then evaluate the parabola at certain points $\boldsymbol{Q}$.

An equivalent formula emphasizes that the parabola passes through $k\boldsymbol{P} = (x_2, y_2)$ rather than through $j\boldsymbol{P} = (x_1, y_1)$:

$$\begin{aligned}\text{parab}(\boldsymbol{X}) &:= (x(\boldsymbol{X}) - x_2)(x(\boldsymbol{X}) + x_2 + x_3 + a_2 + \lambda_1 \lambda_2) \\ &\qquad + (\lambda_1 + \lambda_2 + a_1)(y_2 - y(\boldsymbol{X})) \\ &= (x(\boldsymbol{X}) - x_2)(x(\boldsymbol{X}) + x_2 + x_3 + a_2 + \lambda_1 \lambda_2) \\ &\qquad + (\lambda_1 + \lambda_2 + a_1)y_2 - (\lambda_1 + \lambda_2 + a_1)\, y(\boldsymbol{X}).\end{aligned}$$

It is also possible to expand around the known point $(-2j-k)\boldsymbol{P}$.

The parab($\boldsymbol{X}$) formula is never identically zero (since its $x(\boldsymbol{X})^2$ coefficient is 1) and works well when there are no vertical lines and no point at infinity.

If one uses the parabola of the form

$$\text{parab}(\boldsymbol{X}) := (x(\boldsymbol{X}) - x_1)(x(\boldsymbol{X}) + x_1 + x_3 + a_2 + \lambda_1\lambda_2)$$
$$+ (\lambda_1 + \lambda_2 + a_1)(y_1 - y(\boldsymbol{X})),$$

then one field multiplication suffices to set up the coefficients, provided that λ_1 and λ_2 are already computed.

In some cases it may be more advantageous to multiply out the second half of the equation, such as when one has to evaluate the parabola at complementary points $\boldsymbol{Q}$ and $-\boldsymbol{Q}$. If that is done and the parabola equation

$$\text{parab}(\boldsymbol{X}) := (x(\boldsymbol{X}) - x_1)(x(\boldsymbol{X}) + x_1 + x_3 + a_2 + \lambda_1\ \lambda_2)$$
$$+ (\lambda_1 + \lambda_2 + a_1)\, y_1 - (\lambda_1 + \lambda_2 + a_1)\, y(\boldsymbol{X})$$

is used, then two field multiplications suffice to set up the coefficients. The pairing algorithms should require less computational effort to evaluate a parabola at a point than to take the product of two lines at those points.

One may then obtain a new formula for $f_{2j+k,\boldsymbol{P}}$:

$$f_{2j+k,\boldsymbol{P}}(\boldsymbol{X}) = f_{j,\boldsymbol{P}}(\boldsymbol{X})\, f_{k,\boldsymbol{P}}(\boldsymbol{X})\, f_{j,\boldsymbol{P}}(\boldsymbol{X}) \frac{\text{parab}(\boldsymbol{X})}{\left(x(\boldsymbol{X}) - x_4\right)}.$$

An important saving in this improved technique is that the "parab($\boldsymbol{X}$)" formulae do not reference y_3, so one does not need to compute the y-coordinate of $j\boldsymbol{P}+k\boldsymbol{P}$ if one chooses the first expressions for the slopes which do not involve y_3.

Exemplary tally (not counting the costs for λ_1, λ_2, $(2j+k)\boldsymbol{P}$):

1 multiplication $\lambda_1\lambda_2$ to get coefficients of parab (using first form)

3 multiplications to evaluate parab at $\boldsymbol{Q}$ and $-\boldsymbol{Q}$

(x-coordinate part of computation is shared)

0 to get parab($\boldsymbol{Q}$)/parab($-\boldsymbol{Q}$) as a fraction

0 to get $(x(-\boldsymbol{Q}) - x_4) / (x(\boldsymbol{Q}) - x_4) = 1$

6 multiplications (3 multiplications of fractions) to get

$f_{2j+k,\,\boldsymbol{P}}(\mathbf{Q}) \,/\, f_{2j+k,\,\boldsymbol{P}}(-\mathbf{Q})$ as a fraction

Total 10 field multiplications

The computation of $f_{2j+k,\boldsymbol{P}}$ occurs at some stages of the evaluation of the Weil pairing or the Tate pairing or the squared Weil or Tate pairings for some integers j and k and some point $\boldsymbol{P}$ on the curve. Improving this step thus speeds up all the pairings.

If the characteristic is not equal to 2 or 3, then one can find an equation for the curve such that $a_1=a_2=a_3=0$, and in that case, it is easier to estimate the savings obtained with the improved techniques. Thus, for example, instead of computing two slanted lines and two vertical lines, one need compute only one parabola and a vertical line. The vertical lines are free once the x-coordinates of the points $(j+k)\boldsymbol{P}$ and $(2j+k)\boldsymbol{P}$ are known. Computing two separate slopes for the lines usually requires two inversions and two multiplications in terms of computing power.

One may save even more processing time at the evaluation stage: for each evaluation of $f_{2j+k,\boldsymbol{P}}$ at a point $\boldsymbol{Q}$, where the improved techniques use five multiplications with the parabola, whereas Miller's algorithm would need seven multiplications (assuming that the numerators and denominators are kept track of separately until the end of the computation).

Attention is now drawn to Fig. 3b, which is a flow diagram illustratively depicting an exemplary process 200 in accordance with certain exemplary implementations of present invention. In act 202, at least one curve is determined for use in cryptographically processing selected information. Here, for example, an elliptic curve may be used. In act 204, at least one parabola associated with the elliptic curve is determined. In act 206, pairings are determined using the parabola. In act 208, selected information is cryptographically processed based on the pairing in act 206. Here, the pairings may include Weil pairings, Squared Weil pairings, Tate pairings, Squared Tate pairings, and/or other like pairings.

In certain implementations, the cryptographic processing in act 208 may include either decrypting or encrypting of the selected information and outputting corresponding processed information. By way of example, in certain implementations, process 200 is configured to support key-based cryptography processes, identity-based cryptographic processes, product identification (ID)-based cryptographic processes, short signature-based cryptographic processes, and/or the like.

With acts 204 and 206 at least a first function and a second function that share a point on the elliptic curve can be determined, such that, e.g., the parabola is associated with the shared point, and a first line and a second line associated with the parabola. Act 206 may include determining a third function based on the first line and the second line, and then determining the pairings based on the third function.

The above techniques may be implemented through various forms of logic, including, for example, a programmed computer. Hence, Fig. 4 illustrates a more general exemplary computer environment 400, which can be used in various implementations of the invention. The computer environment 400 is only one example of a computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the computer and network architectures.

Neither should the computer environment 400 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary computer environment 400.

Computer environment 400 includes a general-purpose computing device in the form of a computer 402. Computer 402 can implement, for example, encryptor 102 or decryptor 104 of Fig. 1, generator 120 or client computer 132 of Fig. 2, either or both of modules 152 and 153 of Fig. 3a, and so forth. Computer 402 represents any of a wide variety of computing devices, such as a personal computer, server computer, hand-held or laptop device, multiprocessor system, microprocessor-based system, programmable consumer electronics (e.g., digital video recorders), gaming console, cellular telephone, network PC, minicomputer, mainframe computer, distributed computing environment that include any of the above systems or devices, and the like.

The components of computer 402 can include, but are not limited to, one or more processors or processing units 404, a system memory 406, and a system bus 408 that couples various system components including the processor 404 to the system memory 406. The system bus 408 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, such architectures can include an Industry Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA) local bus, and a Peripheral Component Interconnects (PCI) bus also known as a Mezzanine bus.

Computer 402 typically includes a variety of computer readable media. Such media can be any available media that is accessible by computer 402 and includes both volatile and non-volatile media, removable and non-removable media.

The system memory 406 includes computer readable media in the form of volatile memory, such as random access memory (RAM) 410, and/or non-volatile memory, such as read only memory (ROM) 412. A basic input/output system (BIOS) 414, containing the basic routines that help to transfer information between elements within computer 402, such as during start-up, is stored in ROM 412. RAM 410 typically contains data and/or program modules that are immediately accessible to and/or presently operated on by the processing unit 404.

Computer 402 may also include other removable/non-removable, volatile/non-volatile computer storage media. By way of example, Fig. 4 illustrates a hard disk drive 416 for reading from and writing to a non-removable, non-volatile magnetic media (not shown), a magnetic disk drive 418 for reading from and writing to a removable, non-volatile magnetic disk 420 (e.g., a "floppy disk"), and an optical disk drive 422 for reading from and/or writing to a removable, non-volatile optical disk 424 such as a CD-ROM, DVD-ROM, or other optical media. The hard disk drive 416, magnetic disk drive 418, and optical disk drive 422 are each connected to the system bus 408 by one or more data media interfaces 425. Alternatively, the hard disk drive 416, magnetic disk drive 418, and optical disk drive 422 can be connected to the system bus 408 by one or more interfaces (not shown).

The disk drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data structures, program modules, and other data for computer 402. Although the example illustrates a hard disk 416, a removable magnetic disk 420, and a removable optical disk 424, it is to be appreciated that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes or other magnetic storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or other optical storage, random access memories (RAM), read only memories (ROM), electrically erasable programmable read-only memory (EEPROM), and the like, can also be utilized to implement the exemplary computing system and environment.

Any number of program modules can be stored on the hard disk 416, magnetic disk 420, optical disk 424, ROM 412, and/or RAM 410, including by way of example, an operating system 426, one or more application programs 428, other program modules 430, and program data 432. Each of such operating system 426, one or more application programs 428, other program modules 430, and program data 432 (or some combination thereof) may implement all or part of the resident components that support the distributed file system.

A user can enter commands and information into computer 402 via input devices such as a keyboard 434 and a pointing device 436 (e.g., a "mouse"). Other input devices 438 (not shown specifically) may include a microphone, joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and other input devices are connected to the processing unit 404 via input/output interfaces 440 that are coupled to the system bus 408, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB).

A monitor 442 or other type of display device can also be connected to the system bus 408 via an interface, such as a video adapter 444. In addition to the monitor 442, other output peripheral devices can include components such as speakers (not shown) and a printer 446 which can be connected to computer 402 via the input/output interfaces 440.

Computer 402 can operate in a networked environment using logical connections to one or more remote computers, such as a remote computing device 448. By way of example, the remote computing device 448 can be a personal computer, portable computer, a server, a router, a network computer, a peer device or other common network node, and the like. The remote computing device 448 is illustrated as a portable computer that can include many or all of the elements and features described herein relative to computer 402.

Logical connections between computer 402 and the remote computer 448 are depicted as a local area network (LAN) 450 and a general wide area network (WAN) 452. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When implemented in a LAN networking environment, the computer 402 is connected to a local network 450 via a network interface or adapter 454. When implemented in a WAN networking environment, the computer 402 typically includes a modem 456 or other means for establishing communications over the wide network 452. The modem 456, which can be internal or external to computer 402, can be connected to the system bus 408 via the input/output interfaces 440 or other appropriate mechanisms. It is to be appreciated that the illustrated network connections are exemplary and that other means of establishing communication link(s) between the computers 402 and 448 can be employed.

In a networked environment, such as that illustrated with computing environment 400, program modules depicted relative to the computer 402, or portions thereof, may be stored in a remote memory storage device. By way of example, remote application programs 458 reside on a memory device of remote computer 448. For purposes of illustration, application programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of the computing device 402, and are executed by the data processor(s) of the computer.

Computer 402 typically includes at least some form of computer readable media. Computer readable media can be any available media that can be accessed by computer 402. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media include, but are not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other media which can be used to store the desired information and which can be accessed by computer 402. Communication media typically embody computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media include wired media such as wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

The invention has been described herein in part in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Typically the functionality of the program modules may be combined or distributed as desired in various implementations.

For purposes of illustration, programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of the computer, and are executed by the data processor(s) of the computer.

Alternatively, the invention may be implemented in hardware or a combination of hardware, software, smartcard, and/or firmware. For example, one or more application specific integrated circuits (ASICs) could be designed or programmed to carry out the invention.

Conclusion

Although the description above uses language that is specific to structural features and/or methodological acts, it is to be understood that the invention defined in the appended claims is not limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the invention.